AMENDMENTS TO THE SPECIFICATION:

Please amend paragraphs 0002 and 0003 as follows:

[0002] For many years there have been systems for distributing files within groups of computers, typically computer networks. A typical arrangement involves the provision of a server within the network which allows access to [[filed]]files stored therein by a plurality of computers connected in the network. The files are typically stored in the server and managed thereby.

[0003] Another form of file distribution creates replicas of files onto individual computers within a network. The replicas can be worked [[upon]]on at the individual computers. Replication improves the performance and availability in distributed networks by allowing access to the data even when some of the replicas or network links are not available. Performance is also improved by allowing access to nearby replicas and thereby avoiding expensive remote network access and can also improve throughput of work by allowing a plurality of sites able to work upon data simultaneously. Traditionally, replication has been used in local area networks to ensure the availability of critical functions, including database systems and files. The evolution of mobile computers, including personal digital assistants (PDAs) and remote distributed systems provided by, for example, the Internet, makes replication almost a necessity.

Please amend paragraph 0005 as follows:

[0005] The existing replication algorithms can be said to fall within one of two categories. The first, sometimes known as "pessimistic replication algorithms" relies [[upon]]on single copy usability in which the general concept is to prohibit access to a replica unless it can be proven to be fully up-to-date. This is generally done by categorising one replica as a primary replica which is then responsible for managing the request for a particular object. All the other sites (computers) within the group act as

secondary sites which are only entitled to receive changes from the primary site, that is the primary replica. Other systems serialise the updates by locking all replicas and updates them only at the same time.

Please amend paragraphs 0046 and 0047 as follows:

[0046] FIG. 2 shows an embodiment of distributed file system architecture set up in two computers denoted peer 28 peer 30. The peers 28, 30 may be any two of the computers 10-22 shown in FIG. 1. Each peer 28, 30 is provided with a series of shared files 32, being typically located in a shared file directory. In addition, each peer 28, 30 is provided with a resolver algorithm 34 for handling conflict resolution; a synchronisation algorithm 36 for synchronising file replicas; a reconciliation algorithm 38 for reconciling file versions between computers; a messaging protocol 40 for transferring messages between the computers 28, 30 via a messaging channel 42 and a channel protocol 44 operable, in the embodiment described, via an IP Multicast channel 46. All these elements are described in detail herein.

[0047] The example of architecture shown in FIG. 2 is provided within the distributed file system algorithm stored within each computer 10-22 of the network. The architecture of this embodiment uses a communication channel 46 which provides for communication from one computer to all of the other computers within the network at that point in time. This embodiment uses IP Multicast as a transport layer on which to construct a communication channel. Although this makes for a channel which can sometimes lose information, this is not an issue in the embodiments described because each computer is able eventually to acquire the data it needs. Indeed the channel protocol 44 is able to tolerate three types of channel unreliability: (limited) loss, (finite) duplication and reordering (all of which are inherent to the IP Multicast protocol). This particular messaging protocol provides message buffering, chunking and reconstruction as well as packet duplication and reordering at the lower level. Persons having knowledge of the IP Multicast protocol will know that it includes a congestion control mechanism based on

carrier-sensing and random back-off times which reduces the probability of broadcast collisions and saturation of the Multicast channel. The higher level of this channel sees a standard lossy first-in-first-out (FIFO) one-to-all broadcast channel with buffering in the form of input and output queues.

Please amend paragraph 0065 as follows:

[0065] If it is determined at step 54 that a journal message has been ~~received~~ sent within the last predetermined period, the routine returns to step 50 to continue listening for incoming messages transmitted by other computers 10-22 in the network. On the other hand, if at step 54 it is determined that the computer has not sent any journal message within the last x minutes, the routine proceeds to step 56 at which it is determined whether or not the broadcast channel is clear. If the broadcast channel is not clear, the routine passes to step 58 which provides a delay of a random number of seconds. The routine then passes again to step 56 and, once it is determined that the broadcast channel is clear, the routine passes to step 58 at which the journal message is transmitted from the computer to any other computer which may be networked thereto. The routine then passes back to step 50.

Please amend paragraph 0092 as follows:

[0092] FIG. 8 shows an example of shared file directory system for managing shared files and file transfers. The agent 100, which is typically stored in a folder [[100]] of a computer operating system, provides a shared file directory 102 in which shared files and any shared sub-directories are stored. This shared directory can be accessed directly from software applications of the computer. For example, for a word processing document, it is simply necessary for the user to open the word processing software application and to look for the file to open through the normal file opening procedure. As the file distribution system operates automatically in the background it is not necessary for the user to invoke the file distribution agent when wishing to access this shared file.